NOTES

Notes on *Starting FORTH* for the fig-FORTH User

A very popular book on the FORTH language called *Starting FORTH* has recently been published. The author, Leo Brodie, gives an excellent description of the FORTH language as implemented at FORTH, Inc. fig-FORTH differs from that implementation in some areas, and this document explains those differences. All comments that apply to fig-FORTH also apply to valFORTH.

BLANK = BLANKS (page 285)

Brodie describes the word BLANK. In fig-FORTH, this word is BLANKS.

EMPTY-BUFFERS vs. EMPTY-BUFFERS (page 283)

Brodie's word EMPTY-BUFFERS does not necessarily change the buffers. In fig-FORTH, EMPTY-BUFFERS zero fills the buffers.

CONTEXT vs. CONTEXT (page 247)

These two words are not synonymous in the two versions. fig-FORTH uses a system of VOC-LINKS with CONTEXT, while FORTH, Inc. does not.

EXIT = ;S (page 246)

The word EXIT, as Brodie describes it, is identical in function to ;S in fig-FORTH.

'S = SP@ (page 247)

The word 'S in FORTH, Inc.'s is SP@ in fig-FORTH.

EMPTY (page 84)

Not yet implemented in fig-FORTH.

WIPE vs. CLEAR (page 84)

CLEAR requires a screen number while WIPE clears the last screen edited.

ABORT" (page 103)

Not implemented in fig-FORTH.

?DUP = -DUP (page 103)

The word ?DUP in FORTH, Inc.'s is -DUP in fig-FORTH.

?STACK vs. ?STACK (page 103)

?STACK as described by Brodie is incorrect for fig-FORTH. ?STACK in fig-FORTH automatically aborts if there is a stack error.

NEGATE = MINUS, DNEGATE = DMINUS (pages 123, 178)

The words NEGATE and DNEGATE in FORTH, Inc.'s are MINUS and DMINUS respectively in fig-FORTH.

+LOOP vs. +LOOP (Page 143)

The word +LOOP, as Brodie describes it, works differently for negative stepping than the +LOOP in fig-FORTH. fig-FORTH always ends if the index equals the limit, even for negative stepping.

PAGE = CLS (page 143)

Brodie's PAGE is called CLS in valFORTH. It has no equivalent in fig-FORTH.

U/MOD = U/ (page 177)

Brodie's U/MOD is U/ in fig-FORTH.

CREATE vs. CREATE (page 209)

Brodie's CREATE works differently from CREATE in fig-FORTH. A word using CREATE in fig-FORTH must unSMUDGE the header before the word can be used. The ";" unsmudges headers automatically. In addition, Brodie's CREATE and fig-FORTH CREATE move different default values in the CFA of the created header (see below).

CREATE = <BUILDS (page 209)

In Brodie's chapter 11 on extending the compiler, he uses the series CREATE...DOES>. In fig-FORTH, this should be <BUILDS...DOES>.

NUMBER vs. NUMBER (page 285)

Brodie's NUMBER only converts numbers to double length if the double word set is loaded. fig-FORTH always converts numbers to double length.

>IN = IN, H = DP (page 247)

The variable >IN and H in Brodie's FORTH are IN and DP respectively in fig-FORTH.

VARIABLE vs. VARIABLE (page 209)

The word VARIABLE, as Brodie describes it, accepts no value from the stack. fig-FORTH, on the other hand, does expect an initialization value from the stack.

' vs. ' (page 215)

These words are not synonymous. ' in Brodie is the same as ' 2- in fig-FORTH (or, more properly, ' CFA.)

PROVIDED THROUGH THE COURTESY OF VALPAR INTERNATIONAL CORPORATION; 3801 East 34th Street, Suite 105, Tucson, Arizona. Further distribution of this public domain document must include this notice.

```
Screen: 100
  0 ( Update: write w/o verify      )
  1
  2 BASE @ HEX ASSEMBLER
  3
  4 LABEL -DSK
  5   AD C, 02 C, 03 C, C9 C, 52 C,
  6   D0 C, 05 C, A9 C, 40 C, 4C C,
  7   HERE 4 + ,  A9 C, 80 C, 8D C,
  8   03 C, 03 C, A9 C, 31 C, 8D C,
  9   00 C, 03 C, A9 C, 07 C, 8D C,
 10   06 C, 03 C, A9 C, 80 C,
 11   ( or 00 C, for Percom ) 8D C,
 12   08 C, 03 C, A9 C, 00 C,
 13   ( or 01 C, for Percom ) 8D C,
 14   09 C, 03 C, 20 C, 59 C, E4 C,
 15   60 C,                      -->
```

```
Screen: 101
  0 ( Update: write w/o verify      )
  1
  2 : VERIFY                  ( f -- )
  3   0# 7 * 50 +
  4   [ ' -DISK 7 + ]
  5   LITERAL C! ;
  6
  7 -DSK ' -DISK 27 + !
  8
  9
 10
 11
 12
 13
 14
 15                                -->
```

```
Screen: 102
  0 ( Update: extended boot         )
  1
  2 VOCABULARY XBOOT IMMEDIATE
  3 XBOOT DEFINITIONS
  4
  5 0 VARIABLE #XDBT
  6 0 VARIABLE OLDSTART
  7
  8 : XDBR/W
  9   >R 7F00 +ORIGIN 0FF #XDBT @
 10   BEGIN DUP 0>
 11   WHILE 1- <ROT 2DUP R R/W
 12    1+ SWAP 80 + SWAP ROT
 13   REPEAT R> 2DROP 2DROP
 14   ( screen# LOAD ) ;
 15                                -->
```

```
Screen: 103
  0 ( Update: extended boot         )
  1
  2 : XDBT
  3   1 XDBR/W OLDSTART @
  4   [ ' ABORT 6 + ] LITERAL !
  5   ABORT ;
  6
  7 : SAVPTCH
  8   0 SWAP U/ SWAP DROP DUP 0FF >
  9   IF 0FF - #XDBT !
 10     [ ' ABORT 6 + ] LITERAL @
 11     OLDSTART !
 12     [ ' XDBT CFA ] LITERAL
 13     [ ' ABORT 6 + ] LITERAL !
 14     0FF 252D EXECUTE 0 XDBR/W
 15     OLDSTART @                 -->
```

```
Screen: 104
  0 ( Update: extended boot         )
  1
  2     [ ' ABORT 6 + ] LITERAL !
  3   ELSE
  4     252D EXECUTE
  5   ENDIF ;
  6
  7 0 24D1 C!
  8 ' NOOP CFA 2511 !
  9 ' SAVPTCH CFA 24E4 !
 10
 11 BASE !
 12
 13
 14 FORTH DEFINITIONS
 15
```

```
Screen: 105
  0 ( Subscription to FORTH Dim.    )
  1
  2
  3
  4 Subscription to FORTH Dimensions
  5 is free with membership in the
  6 FORTH Interest Group (fig) at
  7 $15.00 per year ($27.00 foreign
  8 air).  Write:
  9
 10
 11       FORTH Interest Group
 12       P.O. Box 1105
 13       San Carlos, Ca. 94070
 14
 15
```

```
Scr # 85
  0 ( 6502 Assembler in FORTH       )
  1
  2 : MO
  3   <BUILDS
  4      C,
  5   DOES>
  6      SWAP OD CKMODE 1D CKMODE
  7      SWAP C@ MODE @ OR C,
  8      256< 19 MODE @ <> AND
  9      IF C, ELSE , ENDIF
 10      OD MODE ! ;    ( ABS mode )
 11
 12   00 MO ORA,      20 MO AND,
 13   40 MO EOR,      60 MO ADC,
 14   80 MO STA,      A0 MO LDA,
 15   C0 MO CMP,      E0 MO SBC,   -->
```

```
Scr # 83
  0 ( ValFORTH Video editor   V1.1 )
  1
  2 : SPLIT                    ( -- )
  3   YLOC @ 15 <>
  4   IF CBLANK LNINS BOL DUP 32 +
  5      SWAP XLOC @ CMOVE BOL
  6      32 + XLOC @ ERASE CSHOW
  7   ENDIF ;
  8
  9 : -SPLIT                   ( -- )
 10   YLOC @
 11   IF CBLANK BOL HERE 32 CMOVE
 12      LNDEL UPCUR HERE XLOC @ +
 13      BOL XLOC @ + 32 XLOC @ -
 14      CMOVE CSHOW
 15   ENDIF ;                      -->
```

```
Scr # 89
  0 ( ValFORTH Video editor   V1.1 )
  1        7 -> -SPLIT
  2       30 -> LFCUR    31 -> RTCUR
  3      126 -> RUB     127 -> TAB
  4        9 -> INTGL   155 -> NXTLN
  5      255 -> BYTINS  254 -> BYTDEL
  6      157 -> LNINS   156 -> LNDEL
  7       18 -> BFROT     2 -> <BFROT
  8        3 -> BFCLR    11 -> >BFNXT
  9       20 -> >BFLN     6 -> BFLN>
 10       16 -> PRVSCR   14 -> NXTSCR
 11       27 -> SPLCHR    8 -> CLREOL
 12        1 -> ARROW    21 -> BFRPL
 13       15 -> OOPS     10 -> SPLIT
 14   NOSEL PTCHR
 15   SELEND ;                     -->
```

```
Scr # 26
  0 ( Updates: ok  SPEMIT          )
  1
  2 DCX
  3
  4 : ok                      ( -- )
  5   STATE @ 0=
  6   IF QUIT ENDIF ;   IMMEDIATE
  7
  8 : SPEMIT
  9   766 C@ 1 766 C!
 10   SWAP EMIT 766 C! ;
 11
 12 ' SPEMIT CFA
 13 ' EXPECT 126 + !
 14
 15                               -->
```

```
Scr # 27
  0 ( Updates:  U*  U/             )
  1 HEX HERE
  2  75 C, 00 C, 95 C, 00 C, A9 C,
  3  00 C, 75 C, 01 C, 95 C, 01 C,
  4  60 C,
  5 ' U* 1+ @ 27 + 20EA OVER ! 2+ !
  6
  7 HERE
  8  08 C, 38 C, B5 C, 04 C, F5 C,
  9  00 C, A8 C, B5 C, 05 C, F5 C,
 10  01 C, 90 C, 09 C, 94 C, 04 C,
 11  95 C, 05 C, 28 C, 38 C, 4C C,
 12  HERE 9 +  , 28 C, 90 C, 04 C,
 13  94 C, 04 C, 95 C, 05 C, 4C C,
 14  ' U/ DUP 2A + , 1A + 4C OVER
 15  C! 1+ ! DCX                  -->
```

```
Scr # 28
  0 ( Updates: ?TERMINAL  [FMT]    )
  1
  2 HEX
  3 HERE
  4  A9 C, 08 C, 8D C, D01F ,
  5  4C C, ' ?TERMINAL CFA @ ,
  6  ' ?TERMINAL CFA !
  7 DCX
  8
  9
 10 : (FMT)
 11   PAD 772 ! (FMT) ;
 12
 13
 14
 15
```

(1) 6502 Macro Assembler Bug

A bug in the assembler causes ADC, AND, CMP, EOR, LDA, ORA, SBC, and STA, to incorrectly assemble the "absolute,Y" instruction when the "absolute" address lies in the zero page. The problem is that the assembler assumes a "zero,Y" address mode for these instructions when in fact none exists. To fix this problem, change screen 85 of the valFORTH disk so that it matches the screen shown in the listings.

(2) A Display Formatter Bug

Although a display list can be created without error, the routine which moves the list from the work area to its functional address transfers three bytes too few and so leaves behind the "wait and jump on vertical blank" instruction. The display list will appear to work correctly in many cases, but the bug should be fixed. To make this correction, change the last line of code on screen 49 of the Display Formatter disk to read:

```
DSPEND @ 3 + CMOVE ;
```

(3) Shuffling Too Well

The words SHUFL and CSHUFL on screen 148 of the Utilities/Editor package rearrange a sequence of memory locations so that data never end up in the same location as before the shuffle. While this is interesting, it is not what was intended. For a more normal shuffle, replace the phrase DUP I CHOOSE by DUP I 1+ CHOOSE in lines 5 and 12.

(4) A Bug in the Disk FORMATter

(FMT) which is used by FORMAT crashes the system if used as the first disk access after boot. This is because the ROM routine called by (FMT) leaves 128 (not two as documented by Atari) bytes of "bad sector map" data in memory at the location specified in memory location 772 decimal. Since location 772 contains zero on boot, the bad sector map is written starting at address zero, which is fatal. There are several safe places to point 772. PAD is okay if you really want to examine the data, and/or are willing to sacrifice whatever was at PAD (possibly strings). ROM (at $E000 = 57344 decimal) is okay if you just want to get rid of the bad sector data, but someday

Atari may put RAM there in a new machine, with the old ROM code loaded in from disk, so this may not work forever. Here is the fix for putting the map at PAD:

```
: (FMT)          ( unit -- status )
  PAD 772 ! (FMT) ;
```

This fix should be made before either the valFORTH or valDOS FORMAT command is loaded.

(5) A bug in D<

The word D< fails to work correctly when its two arguments are the negatives of each other (e.g., 20000. and -20000.) The following definition (taken from FORTH DIMENSIONS Volume IV, Number 1) replaces the one found on screen 121 of the general utilities package.

```
: D<              ( d1 d2 -- f )
  ROT 2DUP =
  IF
    <ROT DMINUS D+ 0<
  ELSE
    SWAP < SWAP DROP
  ENDIF
  SWAP DROP ;
```

Note that there are three additional double length words found on screen 120 of the the valGRAPHICS disk. They are DU/MOD , D/MOD , and D/ . Here are the definitions for two additional words to round out the set: (D* was taken from FORTH DIMENSIONS Volume IV, Number 1)

```
: DMOD           ( d1 d2 -- drem )
  D/MOD 2DROP ;

: D*             ( d1 d2 -- d3 )
  OVER 5 PICK U*
  6 ROLL 4 ROLL * +
  2SWAP * + ;
```

(6) Bugs in U* and U/

Both U* and U/ have bugs carried over from the original fig model. In addition, the "fix" published in FORTH Dimensions (vol IV, no. 2) also works incorrectly. If you are using another version of FORTH for the Atari, Apple, or other 6502 microcomputer, you can determine if the bugs have been corrected by making the following two

BSCD> (a1 a2 n ---)
Moves n characters from address a1 to address a2, translating from Atari screen code to ATASCII on the way. Used for reading text directly from the screen (as in the video editor).

>SCD (c --- c)
Translates a single character on stack from ATASCII to screen code.

SCD> (c --- c)
Translates a single character on the stack from screen code to ATASCII.

(16) Notes on valDOS

Before exchanging disks, the CLOSE command (like FLUSH) should always be issued. This prevents updated files from one disk to be accidently flushed to another disk. It would be a good idea to CLOSE often just as a precaution.

Also, the description of the system word (ENDF) on LXIII - 3 fails to point out that a status byte is returned. The stack notation should be "fl# -- f". If (ENDF) returns a 1, the operation was successful, otherwise a zero is returned.

Next, there is a set of file buffers set aside for use with open files. The constant MAXFL contains the maximum number of files which can be open at any one time. If MAXFL is changed, the entire valDOS system must be recompiled. The quan WRKSPC points 128 bytes down from the top of the file buffer area. Currently the file buffer area is stored in the middle of the dictionary (see screen 15). If these buffers are moved, the word TOPOM must be changed to reflect the true top of free memory (i.e., if the buffers are moved to top of memory just below the display list, TOPOM must be changed to point to the bottom of the buffers -- BUFBOT).

Because the integrity of any Disk Operating System is vital, we would greatly appreciate it if any bugs found are reported immediately for correction. Thank you!

IMPORTANT ****** IMPORTANT ****** IMPORTANT ****** IMPORTANT

In order to get your copy of SOFTWARE UPDATE #2, send $1.00 and a stamped, self-addressed envelope to:

valFORTH SOFTWARE UPDATE #2
3801 E. 34th Street, Suite 105
Tucson, Az 85713